Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
Electronic Notes in Theoretical Computer Science 265 (2010) 403–421
www.elsevier.com/locate/entcs

A Nominal Relational Model for Local Store

Rasmus Ejlers Møgelberg [1]

IT University of Copenhagen
Denmark

Abstract

The theory of nominal sets is a theory for names, freshness and binders. It has recently been suggested as a framework for modelling local store because it allows for a more elementary development than the traditional presheaf models. However, when modelling the important principle of relational reasoning for local store all these models use families of relations indexed by relations on store, and thus essentially return to presheaf models on the relational level. In this paper we show how relational reasoning can also be modelled using nominal sets. Building on a model suggested by Pitts and Shinwell we construct a relational model for local store in nominal sets in which types are interpreted as relations. These relational interpretations of types capture, in a single relation for each type, the relational reasoning principle for local store which in previous models was captured using a family of relations for each type. The relational model also demonstrates how the relations constitute a model in their own right, which hopefully means that they can be used to construct better models. Using the relational model we construct a relational parametricity principle for the operation allocating local store, and we show how this implies the relational reasoning principle.

Keywords: local store, denotational semantics, parametric polymorphism, nominal sets

1 Introduction

Most programming languages contain a construction for declaring local store, i.e., store cells that can only be accessed by a specified piece of the program. This restriction of access provides an important information hiding principle: changes to an implementations internal use of local store should not affect the observable behaviour of the program. When constructing tools for reasoning about local store a challenging issue is to express the information hiding principle. One idea that has proved useful is relational reasoning: if two programs are implemented using local store, and we can show that there exists a relation between the local stores preserved by the programs, then the programs should be contextually equivalent. This method of proof goes back at least to [12]. Relational reasoning can be proved sound using either syntactic or denotational methods.

[1] This work was supported by the Danish Agency for Science, Technology and Innovation

doi:10.1016/j.entcs.2010.08.024

Traditionally, denotational models of local store have used presheaf categories [11,16,6,14] interpreting types as families of sets indexed over store shapes, i.e., finite sets of cell names. Recently an alternative approach using a continuation monad on the category of nominal sets [5] has been suggested [17,1]. The advantage of this new approach is that the technical development is simpler, mainly because the exponentials in the category of nominal sets have a simpler description than the Kripke style exponents of presheaves. The two approaches are related, because the category of nominal sets is equivalent to a full subcategory of a presheaf topos [5].

In the nominal sets models of local store [1,2] soundness of the relational reasoning principle is proved by constructing, for each type, a family of relations on the denotations of the type, indexed by relations on store, and verifying that the relations corresponding to identity relations on store are contained in contextual equivalence. The construction uses Kripke semantics as one has to take into account that any program can be called at a later time where the store contains more cells. This technique has been successful in creating a useful tool for showing contextual equivalence, but at the moment these relations seem like a trick for obtaining better equational theories on a given model, and their semantic status is unclear.

This paper shows how relational reasoning can be build into nominal sets, and settle the semantic status of the relations by showing that they form a model in their own right. We start by considering a variant of the model of local store used in [1] and then show how a model of relations capturing relational reasoning for local store can be constructed in a very similar way. The key observation for our construction is that the category of nominal sets used in the first model is equivalent to a category of sets indexed over store shapes. Using sorted nominal sets we construct a category of nominal sets equivalent to a category of sets indexed by relations on store, and use this as basis for the relational model. The resulting model can be seen as a nominal representation of the relational presheaf models of [10,9,3].

We treat ground store only. Other authors (e.g. [1,2]) have considered relational reasoning for more advanced notions of store, but we leave it to future work to extend the techniques here to those cases.

1.1 Parametricity for store access operations

Earlier work in the context of presheaf models [10,9,3] has show a connection between local variables and relational parametricity: a procedure can be called in any extension of the store that it is defined in, and it is relationally parametric in this store extension. In this paper we approach the connection between parametricity and local variables from another angle, by showing that the relational reasoning principle for local store follows from a principle of relational parametricity for the operation allocating store cells.

The allocation operation is an algebraic operation [14], and we briefly recall what this means. If T is a (strong) monad, following [8], we think of $T(X)$ as the collection of computations returning values in X. Effectful computations can then arise from algebraic operations or equivalently generic effects [15]. The former are functions that take computations as input and produce computations, e.g. in the

case of store effects we consider operations

$$\mathrm{update}_{\mathbf{b},X} \colon T(X) \to \mathrm{ref}\ \mathbf{b} \to \mathbf{b} \to T(X) \tag{1}$$
$$\mathrm{lookup}_{\mathbf{b},X} \colon (\mathbf{b} \to T(X)) \to \mathrm{ref}\ \mathbf{b} \to T(X) \tag{2}$$

Intuitively, update takes a computation, a reference to a storage cell of type $\mathbf{b}$, a value of type $\mathbf{b}$, and returns the computation that first updates the storage cell with the given value and then continues the original computation. The lookup operation takes a family of computations indexed by values of type $\mathbf{b}$ and a reference to a storage cell, and returns the computation that looks up the value in the storage cell and continues with the corresponding computation.

The algebraic operation corresponding to allocation has type

$$\mathrm{new}_{X,\mathbf{b}} \colon [\mathbb{A}_{\mathbf{b}}]T(X) \to \mathbf{b} \to T(X) \tag{3}$$

For the moment the type $[\mathbb{A}_{\mathbf{b}}]T(X)$ should the thought of as a type of computations in $T(X)$ with a bound name of a reference cell of type $\mathbf{b}$. In Section 3 we shall model this type using atom abstraction in nominal sets. The intuition for $\mathrm{new}_{X,\mathbf{b}}$ is that it returns the computation that allocates a fresh cell of type $\mathbf{b}$, initializes it to the given value of type $\mathbf{b}$ and continues with the given computation with the abstracted name bound to the allocated cell.

The relational model constructed here validates a principle of relational parametricity for $\mathrm{new}_{X,\mathbf{b}}$ in the type of the allocated cell. The crucial ingredient in formulating this principle is a relational interpretation of atom abstraction: for any relation $\mathbf{r} \colon \mathrm{Rel}(\mathbf{b}, \mathbf{b}')$ we construct a relation from $[\mathbb{A}_{\mathbf{b}}]T(X)$ to $[\mathbb{A}_{\mathbf{b}'}]T(X)$, which intuitively relates two computations with bound cell names if they preserve the relation $\mathbf{r}$ (this is to be understood in the sense of the relational reasoning principle for local store). The relational reasoning principle for local store can be seen as a consequence of this parametricity principle.

Section 2 fixes a language with local store, and Section 3 describes a model in nominal sets. Section 4 construct a relational model and shows how this can be used to reason about contextual equivalence. Section 5 describes connections to relational parametricity, and Section 6 gives two examples of using the relational model to show contextual equivalence of programs. Finally Section 7 sketches how a variant of the relational model construction can allow for reasoning using more advanced relations on store.

2 A call-by-value language with local store

We start by fixing a language in which we can write sample programs to illustrate the semantic developments. For this we could have chosen some advanced type theory for effects such as call-by-push-value [6] or the enriched effect calculus [4] and in fact the semantic construction of this paper can be used to construct models of these, but here we choose a simple call-by-value language because it suffices for our purposes.

$$\frac{}{\Theta \mid \Gamma, x\colon \sigma, \Gamma' \vdash^v x\colon \sigma} \qquad \frac{}{\Theta, a\colon \mathbf{b}, \Theta' \mid \Gamma \vdash^v a\colon \mathrm{ref}\ \mathbf{b}}$$

$$\frac{\Theta \mid \Gamma \vdash^v V\colon \sigma}{\Theta \mid \Gamma \vdash^c \mathrm{return}\, V\colon \sigma} \qquad \frac{\Theta \mid \Gamma \vdash^c M\colon \sigma \quad \Theta \mid \Gamma, x\colon \sigma \vdash^c N\colon \tau}{\Theta \mid \Gamma \vdash^c M\,\mathrm{to}\,x.\,N\colon \tau}$$

$$\frac{\Theta \mid \Gamma \vdash^v V\colon \sigma \rightharpoonup \tau \quad \Theta \mid \Gamma \vdash^v W\colon \sigma}{\Theta \mid \Gamma \vdash^c V\,W\colon \tau} \qquad \frac{\Theta \mid \Gamma, x\colon \sigma \vdash^c N\colon \tau}{\Theta \mid \Gamma \vdash^v \lambda x\colon \sigma.\, N\colon \sigma \rightharpoonup \tau}$$

$$\frac{\Theta \mid \Gamma \vdash^c N\colon \tau \quad \Theta \mid \Gamma \vdash^v V\colon \mathbf{b} \quad \Theta \mid \Gamma \vdash^v W\colon \mathrm{ref}\ \mathbf{b}}{\Theta \mid \Gamma \vdash^c W := V; N\colon \tau}$$

$$\frac{\Theta \mid \Gamma, x\colon \mathbf{b} \vdash^c N\colon \tau \quad \Theta \mid \Gamma \vdash^v W\colon \mathrm{ref}\ \mathbf{b}}{\Theta \mid \Gamma \vdash^c \mathrm{read}_W\,\mathrm{as}\,x.\,N\colon \tau}$$

$$\frac{\Theta, a\colon \mathbf{b} \mid \Gamma \vdash^c N\colon \tau \quad \Theta \mid \Gamma \vdash^v V\colon \mathbf{b}}{\Theta \mid \Gamma \vdash^c \mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\,(V)\,\mathrm{in}\, N\colon \tau}$$

Fig. 1. Typing rules for the Fine-Grained-CBV calculus with local store

The language we consider here is a variant of Fine-Grained CBV [7] (Fine-Grained call-by-value) to which we have added local store. We assume given a set Σ of base types and we use $\mathbf{b}$ to range over elements of Σ. The types are given by

$$\sigma, \tau ::= \mathbf{b} \mid \mathrm{ref}\ \mathbf{b} \mid \sigma \rightharpoonup \tau$$

The type ref $\mathbf{b}$ is a type of references to cells of type $\mathbf{b}$. In this paper we only consider references to base types. The type $\sigma \rightharpoonup \tau$ is a call-by-value function space; elements take values as input and produce computations.

Rather than giving an operational semantics we give an equality theory on terms to be considered as an approximation of contextual equality in the spirit of [13]. Because of the presence of effects it is important to specify evaluation order in the language, and for this the language distinguishes between (pure) values terms and computations terms, each of which has its own typing judgement written as

$$\Theta \mid \Gamma \vdash^v V\colon \sigma \qquad\qquad \Theta \mid \Gamma \vdash^c M\colon \sigma$$

respectively. Only computations are allowed to access store.

Typing rules can be found in Figure 1. Open terms have two contexts. The first is a context of variables $x\colon \mathsf{A}$ in the usual sense for which we use Γ as a metavariable. Variables in this context should be thought of as place holders for values. The second context is one of names $a\colon \mathbf{b}$ for which we use Θ as a metavariable. This should be thought of as a list of references to previously allocated distinct cells in the store.

Figure 2 presents an equality theory on terms of Fine-Grained-CBV with local

$$
\begin{aligned}
\mathrm{return}\, V \,\mathrm{to}\, x.\, N &= N[V/x] \\
M \,\mathrm{to}\, x.\, \mathrm{return}\, x &= M \\
(M \,\mathrm{to}\, x.\, N) \,\mathrm{to}\, y.\, P &= M \,\mathrm{to}\, x.\, (N \,\mathrm{to}\, y.\, P) \\
\lambda x{:}\sigma.\, M(V) &= M[V/x] \\
\lambda x{:}\sigma.\, (V\, x) &= V \\
(\mathrm{read}_W \,\mathrm{as}\, z.\, M) \,\mathrm{to}\, x.\, N &= \mathrm{read}_W \,\mathrm{as}\, z.\, (M \,\mathrm{to}\, x.\, N) \\
(W := V; M) \,\mathrm{to}\, x.\, N &= W := V; (M \,\mathrm{to}\, x.\, N) \\
(\mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\, (V) \,\mathrm{in}\, M) \,\mathrm{to}\, x.\, N &= \mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\, (V) \,\mathrm{in}\, (M \,\mathrm{to}\, x.\, N) \\
\mathrm{read}_W \,\mathrm{as}\, x.\, (W := x; M) &= M \\
\mathrm{read}_W \,\mathrm{as}\, x.\, \mathrm{read}_W \,\mathrm{as}\, y.\, M &= \mathrm{read}_W \,\mathrm{as}\, x.\, M[x/y] \\
W := V; W := V'; M &= W := V'; M \\
W := V; \mathrm{read}_W \,\mathrm{as}\, x.\, M &= W := V; M[V/x] \\
\mathrm{read}_a \,\mathrm{as}\, x.\, \mathrm{read}_{a'} \,\mathrm{as}\, y.\, M &= \mathrm{read}_{a'} \,\mathrm{as}\, y.\, \mathrm{read}_a \,\mathrm{as}\, x.\, M \\
a := V; a' := V'; M &= a' := V'; a := V; M \\
a := V; \mathrm{read}_{a'} \,\mathrm{as}\, x.\, M &= \mathrm{read}_{a'} \,\mathrm{as}\, x.\, a := V; M \\
\mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\, (V) \,\mathrm{in}\, (a := V'; M) &= \mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\, (V') \,\mathrm{in}\, M \\
\mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\, (V) \,\mathrm{in}\, (\mathrm{read}_a \,\mathrm{as}\, x.\, M) &= \mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\, (V) \,\mathrm{in}\, M[V/x] \\
\mathrm{let}\, a = \mathrm{new}\, (V) \,\mathrm{in}\, (\mathrm{let}\, a' = \mathrm{new}\, (V') \,\mathrm{in}\, M) &= \mathrm{let}\, a' = \mathrm{new}\, (V') \,\mathrm{in}\, (\mathrm{let}\, a = \mathrm{new}\, (V) \,\mathrm{in}\, M) \\
\mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\, (V) \,\mathrm{in}\, (W := V'; M) &= W := V'; \mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\, (V) \,\mathrm{in}\, M \\
\mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\, (V) \,\mathrm{in}\, (\mathrm{read}_W \,\mathrm{as}\, x.\, M) &= \mathrm{read}_W \,\mathrm{as}\, x.\, \mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\, (V) \,\mathrm{in}\, M \\
\mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\, (V) \,\mathrm{in}\, M &= M\,, \qquad a \notin \mathrm{FV}(M)
\end{aligned}
$$

Fig. 2. The Fine-Grained-CBV calculus. The equality rules are subject to the usual conventions on free variables. Moreover, all rules assume a and a' distinct and the last rule requires a not free in M.

store. The equality theory consist of three different elements: the first five rules are equality rules of Fine-Grained-CBV [7], the next three are algebraicity axioms [15], and the final 13 axiom schemes are the Plotkin-Power axioms for local store [14].

The axioms of Figure 2 provide basic axioms for local store, but do not capture the information hiding principle sketched in the introduction. To illustrate what is needed we give an example of two programs that should be contextually equivalent, but cannot be proved so using the axioms. In Section 6 we show how to prove this using the relational reasoning tools developed in this paper. We stress, however, that this is a standard example of programs that have been proved contextually equivalent, e.g. in [1], but the example is included here for illustration.

Consider the two counters defined as

$$
\begin{aligned}
&\mathrm{let}\, a\colon \mathrm{Int} = \mathrm{new}\, (0) \,\mathrm{in}\, \mathrm{return}\, (\lambda x\colon \mathrm{Int}.\, \mathrm{read}_a \,\mathrm{as}\, z.\, a := z + x; \mathrm{return}\, (z + x)) \\
&\mathrm{let}\, a\colon \mathrm{Int} = \mathrm{new}\, (0) \,\mathrm{in}\, \mathrm{return}\, (\lambda x\colon \mathrm{Int}.\, \mathrm{read}_a \,\mathrm{as}\, z.\, a := z - x; \mathrm{return}\, (x - z))
\end{aligned}
$$

Both of these are computations of type $\mathrm{Int} \rightharpoonup \mathrm{Int}$ which compute functions, that

given an integer x return the sum of the integers by which they have been called previously. They also both accomplish this task by allocating a local store cell, but where the first counter keeps the sum of the previous x's in that local cell, the second counter keeps the negative of this number.

To prove these programs contextually equivalent one usually uses relational reasoning: consider the relation on stores relating s to s' if $s(a) = -s'(a)$. If we imagine the two counters being run on parallel machines with stores in this relation, then each pair of calls to the counters with equal inputs not only gives equal outputs, but also leaves the stores related. Using the techniques of [12,1] this suffices for showing the programs equivalent.

3 A model in nominal sets

We construct an interpretation of the Fine-Grained CBV calculus with local store using nominal sets [5]. The interpretation is a variant of the one used in [17,1]. We assume that we are given for each $\mathbf{b} \in \Sigma$ an interpretation of $\mathbf{b}$ as a set $[\![\mathbf{b}]\!]$. Moreover we assume that we are given a *sorted collection of atoms* by which we mean a map of sets $\mathbb{A} \to \Sigma$, such that for each sort $\mathbf{b}$ there are infinitely many atoms mapping to $\mathbf{b}$ by the sorting function. We shall think of a sorted collection of atoms as a signature for a typed store. The set $\mathbb{A}$ is the set of cells and the sorting maps a cell to the type of the content stored in that cell. The set of atoms with sort $\mathbf{b}$ is denoted $\mathbb{A}_{\mathbf{b}}$, and we often write $\mathbf{a}_{\mathbf{b}}$ to indicate that $\mathbf{a}$ is an atom of sort $\mathbf{b}$.

Given a sorted collection of atoms $\mathbb{A} \to \Sigma$ we can form the category of nominal sets, which we denote $\mathbf{Nom}$ or sometimes $\mathbf{Nom}_{[\mathbb{A}\to\Sigma]}$ when the atom sorting is not clear from context, as usual. First we consider the group $\mathbf{Perm}(\mathbb{A} \to \Sigma)$ of finite permutations (i.e. permutations fixing all but a finite number of elements) of $\mathbb{A}$ respecting sortings. The category $\mathbf{Nom}$ has as objects sets with a left $\mathbf{Perm}(\mathbb{A} \to \Sigma)$-action, ie., a map $\cdot\colon \mathbf{Perm}(\mathbb{A} \to \Sigma) \times X \to X$ such that $(\pi \circ \pi') \cdot x = \pi \cdot (\pi' \cdot x)$ and $id \cdot x = x$ such that each x in X has a finite support, i.e. a finite set A of atoms such that if π fixes all elements of A then $\pi \cdot x = x$. It is well known that $\mathbf{Nom}$ is a cartesian closed category with exponent given by the set of finitely supported functions with respect to the action defined by $(\pi \cdot f)(x) = \pi \cdot (f(\pi^{-1} \cdot x))$, see [5]. We write $A \times B$ and $A \to_{\mathrm{fs}} B$ or B^A for the cartesian closed structure on the category of nominal sets and reserve $A \to B$ for the set of functions with empty support between a pair of nominal sets. We write $x \in X$ for X a nominal set if x is an element in the underlying set of X.

As in [17,1] we use a cps style semantics, because it greatly simplifies the interpretation of allocation. So we assume given a nominal set R of results, which has trivial permutation action. The set $\mathbb{A}$ is a nominal set with the obvious action and recalling that each sort $\mathbf{b}$ has an interpretation as a set, we can consider each of these a nominal set with trivial permutation action.

We define the nominal set of stores as

$$\mathbb{S} = \textstyle\prod_{\mathbf{b}\in\Sigma} \mathbb{A}_{\mathbf{b}} \to_{\mathrm{fs}} [\![\mathbf{b}]\!].$$

$$\begin{aligned}
[\![\mathrm{read}_W \,\mathrm{as}\, x.\, N]\!]_\rho(k)(s) &= [\![N]\!]_{\rho[x \mapsto s([\![W]\!]_\rho)]}(k)(s) \\
[\![W := V; N]\!]_\rho(k)(s) &= [\![N]\!]_\rho(k)(s[[\![W]\!]_\rho \mapsto [\![V]\!]_\rho]) \\
[\![\mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\,(V)\,\mathrm{in}\, N]\!]_\rho(k)(s) &= \mathrm{fresh}\, \mathbf{a_b}.\ [\![N]\!]_{\rho[a \mapsto \mathbf{a_b}]}(k)(s[\mathbf{a_b} \mapsto [\![V]\!]_\rho])
\end{aligned}$$

Fig. 3. Interpretation of store access operations.

Note that by this product we mean the product in **Nom**, i.e., we only consider elements f with a finite set of atoms which supports all of the $f_{\mathbf{b}} \colon \mathbb{A}_{\mathbf{b}} \to_{\mathrm{fs}} [\![\mathbf{b}]\!]$. We usually assume that all the sets $[\![\mathbf{b}]\!]$ are non-empty as otherwise $\mathbb{S}$ becomes empty.

A continuation is a map that takes a store and produces a result in R, however, by our definition of $\mathbb{S}$, stores are infinite and intuitively, computable continuations can only look at finite subsets of the store. The restriction to finitely supported functions is not sufficient to ensure this restriction, as, e.g., maps counting the number of cells holding a specific value are finitely supported (in fact equivariant). Instead we define our collection of continuations to be the nominal set

$$\mathbb{K} = \{k\colon R^{\mathbb{S}} \mid \exists A \subset_{\mathrm{fin}} \mathbb{A}.\, \forall s, s' \in \mathbb{S}.\, (\forall \mathbf{a} \in A.\, s(\mathbf{a}) = s'(\mathbf{a})) \Longrightarrow k(s) = k(s')\}\,. \quad (4)$$

This restriction has important consequences for the model and is also crucial for the construction of the relational model.

Lemma 3.1 *If $k \in \mathbb{K}$ and $s, s' \in \mathbb{S}$ agree on the support of k, then $k(s) = k(s')$.*

We interpret Fine-Grained CBV using the monad $T = \mathbb{K}^{\mathbb{K}^{(-)}}$ on **Nom**. The type ref $\mathbf{b}$ is interpreted as $\mathbb{A}_{\mathbf{b}}$ and $\sigma \rightharpoonup \tau$ is interpreted as $[\![\sigma]\!] \to T([\![\tau]\!])$. Value terms $\Theta \mid \Gamma \vdash^v V : \sigma$ are interpreted as equivariant maps $[\![\Theta]\!] \times [\![\Gamma]\!] \to [\![\sigma]\!]$ and computation terms $\Theta \mid \Gamma \vdash^c V : \sigma$ are interpreted as equivariant maps $[\![\Theta]\!] \times [\![\Gamma]\!] \to T([\![\sigma]\!])$. The context Γ is interpreted as usual using products, but following the intuition given in Section 2, a name context $\Theta = a_1\colon \mathbf{b}_1, \ldots, a_n\colon \mathbf{b}_n$ is interpreted as the set $\otimes_{\mathbf{b}_i \in \overline{\mathbf{b}}} \mathbb{A}_{\mathbf{b}_i}$ of tuples $(\mathbf{a}^1, \ldots, \mathbf{a}^n)$ of *distinct* elements with $\mathbf{a}^i \in \mathbb{A}_{\mathbf{b}_i}$. This is a special case of the tensor product on **Nom**:

$$A \otimes B = \{(x, y) \mid x \sharp y\} \quad (5)$$

where $x \sharp y$ means that x, y have disjoint support.

We omit the details of the standard monadic interpretation and focus on the interpretation of the operations on store. The interpretations are given in Figure 3. In the figure we use environments ρ which are maps specifying elements in $[\![\Theta]\!] \times [\![\Gamma]\!]$ in the hopefully obvious way. Also the notation $s[\mathbf{a_b} \mapsto n]$ is used for updating s at location $\mathbf{a_b}$, where $s \in \mathbb{S}$, i.e., $s[\mathbf{a_b} \mapsto n](\mathbf{a'_b})$ is n if $\mathbf{a'_b} = \mathbf{a_b}$ and $s(\mathbf{a'_b})$ otherwise. We have used similar notation for updating the environment ρ.

In the interpretation of cell allocation we have used the notation $\mathrm{fresh}\, \mathbf{a_b}$. This means: pick some $\mathbf{a_b}$ not in the support of k, s, ρ and compute the element to the right of the fresh statement. Of course, for this to be well defined, one must show

that the element on the right is independent of the particular choice of $\mathbf{a_b}$, which can be done using standard techniques from nominal set theory.

Proposition 3.2 *The model validates the axioms of Figure 2.*

The last axiom of Figure 2 (garbage collection) does not hold in the model of [1], but can only be verified up to relational reasoning. The reason it holds in this model is the restriction to continuations in $\mathbb{K}$.

Proof. We verify the garbage collection axiom. The condition of a not free in M means that $\Theta \mid \Gamma \vdash^c M : \sigma$ is well typed. A simple induction shows that $[\![\Theta \mid \Gamma \vdash^c M : \sigma]\!]_\rho = [\![\Theta, a : \mathbf{b} \mid \Gamma \vdash^c M : \sigma]\!]_{\rho[a \mapsto \mathbf{a_b}]}$. In particular this means that if $\mathbf{a_b}$ is fresh for ρ and k then it is also fresh for $[\![\Theta, a : \mathbf{b} \mid \Gamma \vdash^c M : \sigma]\!]_{\rho[a \mapsto \mathbf{a_b}]}(k)$ which is an element in $\mathbb{K}$. We compute

$$\begin{aligned}
&[\![\text{let } a : \mathbf{b} = \text{ new } (V) \text{ in } M]\!]_\rho(k)(s) \\
&\quad = \text{fresh } \mathbf{a_b}.\ [\![\Theta, a : \mathbf{b} \mid \Gamma \vdash^c M : \sigma]\!]_{\rho[a \mapsto \mathbf{a_b}]}(k)(s[\mathbf{a_b} \mapsto [\![V]\!]_\rho]) \\
&\quad = \text{fresh } \mathbf{a_b}.\ [\![\Theta, a : \mathbf{b} \mid \Gamma \vdash^c M : \sigma]\!]_{\rho[a \mapsto \mathbf{a_b}]}(k)(s) \\
&\quad = [\![\Theta \mid \Gamma \vdash^c M : \sigma]\!]_\rho(k)(s)\,.
\end{aligned}$$

The equality between line 2 and 3 is a consequence of Lemma 3.1. □

3.1 Algebraic operations for local store

The interpretation of local store in Figure 3 can be read as defining algebraic operations for local store for the monad T in the sense of Plotkin and Power [14]. The types of these were suggested in the introduction (1, 2, 3), but in fact, we can give these more general types, defining e.g.,

$$\begin{aligned}
\text{update}_{\mathbf{b},X} &: \mathbb{K}^X \to \mathbb{A}_\mathbf{b} \to_{\text{fs}} [\![\mathbf{b}]\!] \to_{\text{fs}} \mathbb{K}^X \\
\text{lookup}_{\mathbf{b},X} &: ([\![\mathbf{b}]\!] \to_{\text{fs}} \mathbb{K}^X) \to \mathbb{A}_\mathbf{b} \to_{\text{fs}} \mathbb{K}^X
\end{aligned}$$

for all nominal sets X. The operations are defined as

$$\begin{aligned}
\text{update}_{\mathbf{b},X}(f)(\mathbf{a_b})(v) &= \lambda x : X.\ \lambda s : \mathbb{S}.\ f(x)(s[\mathbf{a_b} \mapsto v]) \\
\text{lookup}_{\mathbf{b},X}(f)(\mathbf{a_b}) &= \lambda x : X.\ \lambda s : \mathbb{S}.\ f(s(\mathbf{a_b}))(x)(s)
\end{aligned}$$

As mentioned in the introduction, the algebraic operation for cell allocation takes a computation with a bound cell name as input. Since the result of applying the operation should not depend on the name of the abstracted cell, the appropriate input type is given by atom abstraction [5]. Recall that the atom abstraction $[\mathbb{A}_\mathbf{b}]X$ is defined as the quotient $\mathbb{A}_\mathbf{b} \times X / \sim$ where $(\mathbf{a}, x) \sim (\mathbf{a}', x')$ if $(\mathbf{a}\,\mathbf{a}'')\,x = (\mathbf{a}'\,\mathbf{a}'')\,x'$ for some (indeed any) fresh $\mathbf{a}''$. Recall also that there is a *concretion* map $(-)@(=) : [\mathbb{A}_\mathbf{b}]X \otimes \mathbb{A}_\mathbf{b} \to X$ defined as $[(\mathbf{a}, x)]@\,\mathbf{a}' = (\mathbf{a}\,\mathbf{a}')\,x$. Note that this is only well defined for $\mathbf{a}'$ fresh for $[(\mathbf{a}, x)]$, i.e., for $\mathbf{a}' \notin \text{supp}(x) \setminus \{\mathbf{a}\}$.

Now, the algebraic operation for allocating a fresh storage cell has type

$$\mathrm{new}_{X,\mathbf{b}} \colon [\mathbb{A}_{\mathbf{b}}]\mathbb{K}^X \to [\![\mathbf{b}]\!] \to_{\mathrm{fs}} \mathbb{K}^X$$

and is defined as $\mathrm{new}_{X,\mathbf{b}}(f)(v) = \lambda x\colon X.\, \lambda s\colon \mathbb{S}.\, \mathrm{fresh}\, \mathbf{a}_{\mathbf{b}}.\ (f @ \mathbf{a}_{\mathbf{b}})(x)(s[\mathbf{a}_{\mathbf{b}} \mapsto v])$.

Algebraic operations correspond to generic effects [15]. The update and lookup correspond to maps $\mathbb{A}_{\mathbf{b}} \times [\![\mathbf{b}]\!] \to T(1)$ and $\mathbb{A}_{\mathbf{b}} \to T([\![\mathbf{b}]\!])$ respectively. Following Staton [18], one can describe allocation using a family of generic effects of type

$$\mathrm{alloc}_{\mathbf{b},\otimes_i \mathbb{A}_{\mathbf{b}_i}} \colon\ \otimes_i \mathbb{A}_{\mathbf{b}_i} \times [\![\mathbf{b}]\!] \to T((\otimes_i \mathbb{A}_{\mathbf{b}_i}) \otimes \mathbb{A}_{\mathbf{b}})\,. \tag{6}$$

These can be thought of as maps computing an extended context from a given one.

4 A relational model construction

In this section we construct a relational model of Fine-Grained CBV with local store. The construction is similar to that of Section 3 except that we pass from sets to relations. We do this first for the base types. Suppose that we are given a set of base types Σ and an interpretation of each base type as a set, as in Section 3. Consider the set of relations

$$\Sigma^{\mathrm{Rel}} = \{(\mathbf{b}, \mathbf{b}', R) \mid \mathbf{b}, \mathbf{b}' \in \Sigma, R\colon \mathrm{Rel}([\![\mathbf{b}]\!], [\![\mathbf{b}']\!]),\ R \text{ non-empty}\}\,.$$

The restriction to *non-empty* relations is necessary to get an interesting model (if we omit it, the $\mathbb{K}_{\mathrm{Rel}}$ defined below becomes the total relation, relating all pairs of continuations) and is similar to the restriction that $[\![\mathbf{b}]\!]$ is non-empty. We use $\mathbf{r}$ as a metavariable to range over Σ^{Rel} and also use $\mathbf{r}$ for the relation part of the triple.

We assume that we are given a sorted collection of atoms $p\colon \mathbb{A}^{\mathrm{Rel}} \to \Sigma^{\mathrm{Rel}}$ (we call the sorting p because the main example is the projection $\Sigma^{\mathrm{Rel}} \times \mathbb{N} \to \Sigma^{\mathrm{Rel}}$). To give some intuition for why $\mathbf{Nom}_{[\mathbb{A}^{\mathrm{Rel}} \to \Sigma^{\mathrm{Rel}}]}$ is an interesting category to consider, we recall the relationship between nominal sets and presheaf categories [5]. Suppose we are given a sorted collection of atoms $\mathbb{A} \to \Sigma$ as in Section 3, following the intuition that this is a signature for typed store, we define a store shape to be a finite subset of $\mathbb{A}$. An object in $\mathbf{Nom}_{[\mathbb{A} \to \Sigma]}$ defines a family of sets indexed by store shapes, namely the family $\{x \in X \mid \mathrm{supp}(x) \subseteq A\}_{A \subseteq_{\mathrm{fin}} \mathbb{A}}$.

Similarly, we may consider finite subsets of $\mathbb{A}^{\mathrm{Rel}}$ to be simple finite relations on stores, namely A describes the relation relating s to s' if $\mathbf{r}(s(\mathbf{a}_{\mathbf{r}}), s'(\mathbf{a}_{\mathbf{r}}))$ holds for all $\mathbf{a}_{\mathbf{r}} \in A$. This way, an object of $\mathbf{Nom}_{[\mathbb{A}^{\mathrm{Rel}} \to \Sigma^{\mathrm{Rel}}]}$ can be considered as a family of sets indexed by relations on stores. For relational reasoning for local store we need families of relations indexed over finite relations on stores, and in this paper we shall consider relations in $\mathbf{Nom}_{[\mathbb{A}^{\mathrm{Rel}} \to \Sigma^{\mathrm{Rel}}]}$. The relations on local store described above are extremely simple, but we have chosen this simplicity for presentation purposes. In Section 5 we sketch how to extend this idea to more advanced relations on store.

Following the intuition above it is tempting to define a relation on store objects as follows: writing $\mathbb{S}_{\mathrm{dom}}$ and $\mathbb{S}_{\mathrm{cod}}$ for $\prod_{\mathbf{r} \in \Sigma^{\mathrm{Rel}}} \mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}} \to \mathrm{dom}(\mathbf{r})$ and $\prod_{\mathbf{r} \in \Sigma^{\mathrm{Rel}}} \mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}} \to$

$\mathrm{cod}(\mathbf{r})$ respectively define $\mathbb{S}_{\mathrm{Rel}} \colon \mathrm{Rel}(\mathbb{S}_{\mathrm{dom}}, \mathbb{S}_{\mathrm{cod}})$ by

$$\mathbb{S}_{\mathrm{Rel}}(s, s') \iff \forall \mathbf{a_r} \in \mathbb{A}^{\mathrm{Rel}}. \mathbf{r}(s(\mathbf{a_r}), s'(\mathbf{a_r}))$$

Unfortunately the $\mathbb{S}_{\mathrm{Rel}}$ so defined is empty. However, we can define a relation on continuations as follows. Consider first the continuation objects $\mathbb{K}_{\mathrm{dom}} \subseteq R^{\mathbb{S}_{\mathrm{dom}}}, \mathbb{K}_{\mathrm{cod}} \subseteq R^{\mathbb{S}_{\mathrm{cod}}}$ defined as in Section 3 and define

$$\begin{aligned}\mathbb{K}_{\mathrm{Rel}} = \{(k_d, k_c) \in & \mathbb{K}_{\mathrm{dom}} \times \mathbb{K}_{\mathrm{cod}} \mid \exists A \subseteq_{\mathrm{fin}} \mathbb{A}^{\mathrm{Rel}}. \forall s_d, s_c. \\ & (\forall \mathbf{a_r} \in A. \mathbf{r}(s_d(\mathbf{a_r}), s_c(\mathbf{a_r}))) \Longrightarrow k_d(s_d) = k_c(s_c)\}\end{aligned}$$

The relation $\mathbb{K}_{\mathrm{Rel}}$ satisfies an equivariance property: if π is a finite permutation respecting the atom sorting $\mathbb{A}^{\mathrm{Rel}} \to \Sigma^{\mathrm{Rel}}$ then $\mathbb{K}_{\mathrm{Rel}}(k_d, k_c)$ holds iff $\mathbb{K}_{\mathrm{Rel}}(\pi \cdot k_d, \pi \cdot k_c)$.

Note the similarity of the definition of $\mathbb{K}_{\mathrm{Rel}}$ to (4).

Lemma 4.1 *Let $k_d \in \mathbb{K}_{\mathrm{dom}}, k_c \in \mathbb{K}_{\mathrm{cod}}$ be given. Then $(k_d, k_c) \in \mathbb{K}_{\mathrm{Rel}}$ iff*

$$(\forall \mathbf{a_r} \in \mathrm{supp}(k_d) \cup \mathrm{supp}(k_c). \mathbf{r}(s_d(\mathbf{a_r}), s_c(\mathbf{a_r}))) \Longrightarrow k_d(s_d) = k_c(s_c)$$

for all $s_d \in \mathbb{S}_{\mathrm{dom}}$ and $s_c \in \mathbb{S}_{\mathrm{cod}}$.

At the moment we have several atom sortings in play. Apart from the given one $p \colon \mathbb{A}^{\mathrm{Rel}} \to \Sigma^{\mathrm{Rel}}$ there are the two compositions of p with the domain and codomain map $\mathrm{dom} \circ p, \mathrm{cod} \circ p \colon \mathbb{A}^{\mathrm{Rel}} \to \Sigma$. There is a pair of functors

$$\mathbf{Nom}_{[\mathrm{dom} \circ p]} \longrightarrow \mathbf{Nom}_{[\mathbb{A}^{\mathrm{Rel}} \to \Sigma^{\mathrm{Rel}}]} \longleftarrow \mathbf{Nom}_{[\mathrm{cod} \circ p]},$$

given simply by restriction of permutation action.

We now define the category **Rel** which will be the base for the relational model of local store. Following the intuition given in the beginning of this section, **Rel** can be thought of as a category of relations indexed by relations on stores.

Definition 4.2 The category **Rel** has

Objects: Triples (X, Y, Q) where X, Y are objects of $\mathbf{Nom}_{[\mathrm{dom} \circ p]}$ and $\mathbf{Nom}_{[\mathrm{cod} \circ p]}$ respectively and Q is an equivariant (in the sense of $\mathbf{Nom}_{[\mathbb{A}^{\mathrm{Rel}} \to \Sigma^{\mathrm{Rel}}]}$) subset of $X \times Y$.

Morphisms: A morphism from (X, Y, Q) to (X', Y', Q') is a pair of maps $f \colon X \to X'$ and $g \colon Y \to Y'$ in $\mathbf{Nom}_{[\mathrm{dom} \circ p]}$ and $\mathbf{Nom}_{[\mathrm{cod} \circ p]}$ respectively, such that (f, g) map pairs related in Q to pairs related in Q'.

The triple $(\mathbb{K}_{\mathrm{dom}}, \mathbb{K}_{\mathrm{cod}}, \mathbb{K}_{\mathrm{Rel}})$ defines an object of **Rel** by the equivariance principle noticed above. Any element of Σ^{Rel} defines an object of **Rel** in the obvious way. For any $\mathbf{r}$ we can define a relational interpretation of atoms. Define first (with a slight misuse of notation)

$$\mathbb{A}^{\mathrm{Rel}}_{\mathrm{dom}(\mathbf{r})} = \{\mathbf{a} \mid \mathrm{dom}(p(\mathbf{a})) = \mathrm{dom}(\mathbf{r})\} \tag{7}$$

$$\mathbb{A}^{\mathrm{Rel}}_{\mathrm{cod}(\mathbf{r})} = \{\mathbf{a} \mid \mathrm{cod}(p(\mathbf{a})) = \mathrm{cod}(\mathbf{r})\} \tag{8}$$

with permutation actions given by application. The relation is given (as a span) by the inclusion of $\mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}} = \{\mathbf{a} \mid p(\mathbf{a}) = \mathbf{r}\}$ into both these sets. This defines an object of **Rel** for which we shall often simply write $\mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}}$. In general, we shall usually simply denote objects of **Rel** simply by their third component.

Proposition 4.3 *The category* **Rel** *is cartesian closed and the domain and codomain functors*

$$\mathbf{Nom}_{[\mathrm{dom}\circ p]} \longleftarrow \mathbf{Rel} \longrightarrow \mathbf{Nom}_{[\mathrm{cod}\circ p]} \tag{9}$$

preserve the cartesian closed structure.

Proof. The products of two relations relate pairs whose components are related. The exponent: $(X, Y, Q) \to (X', Y', Q')$ is the triple $(X \to_{\mathrm{fs}} X', Y \to_{\mathrm{fs}} Y', Q'^{Q})$ where the first two components are constructed using exponents in $\mathbf{Nom}_{[\mathrm{dom}\circ p]}$ and $\mathbf{Nom}_{[\mathrm{cod}\circ p]}$ respectively, and $Q'^{Q}(f, g)$ holds if $Q(x, y)$ implies $Q'(f(x), g(y))$. □

The tensor products on $\mathbf{Nom}_{[\mathrm{dom}\circ p]}$ and $\mathbf{Nom}_{[\mathrm{cod}\circ p]}$ (5) extend to a tensor product on **Rel**: the tensor of two relations relates pairs whose components are related.

We now describe how Fine-Grained-CBV can be interpreted in **Rel**. Each base type $\mathbf{b}$ is interpreted as the identity relation $eq_{[\![\mathbf{b}]\!]}$ on the set $[\![\mathbf{b}]\!]$ and ref $\mathbf{b}$ is interpreted as $\mathbb{A}^{\mathrm{Rel}}_{eq_{[\![\mathbf{b}]\!]}}$. The rest of the types are interpreted using the cartesian closed structure on **Rel** and the monad $T^{\mathrm{Rel}} = \mathbb{K}^{(-)}_{\mathrm{Rel}}_{\mathrm{Rel}}$. Since the projections (9) preserve all this structure, the interpretation of a type σ in the relational model is a relation $[\![\sigma]\!]^{\mathrm{Rel}} \colon \mathrm{Rel}([\![\sigma]\!]^{\mathrm{dom}}, [\![\sigma]\!]^{\mathrm{cod}})$ where $[\![\sigma]\!]^{\mathrm{dom}}$ and $[\![\sigma]\!]^{\mathrm{cod}}$ are the interpretations of σ as defined in Section 3 using the atom sortings dom and cod respectively.

Variable contexts Γ can be interpreted using the products in **Rel**. Name contexts $\Theta = a_1 \colon \mathbf{b}_1, \ldots, a_n \colon \mathbf{b}_n$ are interpreted as $\otimes_i \mathbb{A}_{eq_{\mathbf{b}_i}}$. The following "logical relations lemma" gives the interpretation of terms in the relational model.

Proposition 4.4 *If* $\Theta \mid \Gamma \vdash^{v} V : \sigma$ *and* $\Theta \mid \Gamma \vdash^{c} M : \sigma$ *then*

$$([\![V]\!]^{\mathrm{dom}}, [\![V]\!]^{\mathrm{cod}}) \colon [\![\Theta]\!]^{\mathrm{Rel}} \times [\![\Gamma]\!]^{\mathrm{Rel}} \to [\![\sigma]\!]^{\mathrm{Rel}}$$
$$([\![M]\!]^{\mathrm{dom}}, [\![M]\!]^{\mathrm{cod}}) \colon [\![\Theta]\!]^{\mathrm{Rel}} \times [\![\Gamma]\!]^{\mathrm{Rel}} \to T^{\mathrm{Rel}}([\![\sigma]\!]^{\mathrm{Rel}})$$

where $[\![-]\!]^{\mathrm{dom}}, [\![-]\!]^{\mathrm{cod}}$ *denote the interpretations of terms in the model of Section 3 using the atom sortings* dom *and* cod *respectively.*

Proposition 4.4 is proved by induction over the typing rules, but we omit the details. Most cases follow from the fact that the interpretation of Fine-Grained-CBV is given by the cartesian closed structure and the monad, and both projections of (9) preserve this structure. This leaves the operations for local store, and these cases follow from Theorem 5.1 below.

4.1 Approximating contextual equivalence

The semantic relations constructed above define an equality relation on computation terms: if $\Theta \mid \Gamma \vdash^c M, N : \sigma$, we write $M \sim_\sigma N$ if

$$(\llbracket M \rrbracket^{\mathrm{dom}}, \llbracket N \rrbracket^{\mathrm{cod}}) \colon \llbracket \Theta \rrbracket^{\mathrm{Rel}} \times \llbracket \Gamma \rrbracket^{\mathrm{Rel}} \to T^{\mathrm{Rel}}(\llbracket \sigma \rrbracket^{\mathrm{Rel}})$$

In the following we shall show that this defines an equivalence relation on terms and that it is contained in contextual equivalence.

Is the semantic relation $\llbracket \sigma \rrbracket^{\mathrm{Rel}}$ an equivalence relation? At the moment it is not even an endorelation. Under reasonable assumptions however, we can make $\llbracket \sigma \rrbracket^{\mathrm{Rel}}$ an endorelation. Consider the map $(-)^{\mathrm{op}} \colon \Sigma^{\mathrm{Rel}} \to \Sigma^{\mathrm{Rel}}$ which maps a relation to its opposite. For the rest of the paper we shall assume that there is a lift of this map to atoms, i.e., a map making the diagram commute

$$\begin{array}{ccc} \mathbb{A}^{\mathrm{Rel}} & \xrightarrow{(-)^{\mathrm{op}}} & \mathbb{A}^{\mathrm{Rel}} \\ \downarrow{\scriptstyle p} & & \downarrow{\scriptstyle p} \\ \Sigma^{\mathrm{Rel}} & \xrightarrow{(-)^{\mathrm{op}}} & \Sigma^{\mathrm{Rel}} \end{array}$$

such that $((\mathbf{a_r})^{\mathrm{op}})^{\mathrm{op}} = \mathbf{a_r}$, and $\mathbf{a}^{\mathrm{op}}_{eq_\mathbf{b}} = \mathbf{a}_{eq_\mathbf{b}}$. This can be defined for example if $\mathbb{A}^{\mathrm{Rel}} = \Sigma^{\mathrm{Rel}} \times \mathbb{N}$ with sorting map the first projection. Such a lift induces an isomorphism of atom sortings between $\mathrm{dom} \circ p \colon \mathbb{A}^{\mathrm{Rel}} \to \Sigma$ and $\mathrm{cod} \circ p \colon \mathbb{A}^{\mathrm{Rel}} \to \Sigma$, which induces an isomorphism of categories $\mathbf{Nom}_{[\mathrm{dom} \circ p]} \cong \mathbf{Nom}_{[\mathrm{cod} \circ p]}$ by the following general proposition.

Proposition 4.5 *Suppose $\alpha \colon \mathbb{A} \to \mathbb{A}'$ is an isomorphism commuting with the atom storting $\mathbb{A} \to \Sigma$ and $\mathbb{A}' \to \Sigma$. Then $\mathbf{Nom}_{[\mathbb{A} \to \Sigma]}$ and $\mathbf{Nom}_{[\mathbb{A}' \to \Sigma]}$ are isomorphic. The isomorphism maps a nominal set X to X with action defined by $\pi \cdot x = (\alpha^{-1} \circ \pi \circ \alpha) \cdot x$.*

Up to this isomorphism $\llbracket \sigma \rrbracket^{\mathrm{dom}} \cong \llbracket \sigma \rrbracket^{\mathrm{cod}}$, and we can consider $\llbracket \sigma \rrbracket^{\mathrm{Rel}}$ an endorelation on $\llbracket \sigma \rrbracket^{\mathrm{dom}}$, and we shall often simply write $\llbracket \sigma \rrbracket$ for $\llbracket \sigma \rrbracket^{\mathrm{dom}}$.

Theorem 4.6 sums up the main properties of the relations defined above.

Theorem 4.6 (i) *Each relation $\llbracket \sigma \rrbracket^{\mathrm{Rel}}$ is symmetric and zigzag-closed, i.e.,*

$$\llbracket \sigma \rrbracket^{\mathrm{Rel}}(x, y) \wedge \llbracket \sigma \rrbracket^{\mathrm{Rel}}(z, y) \wedge \llbracket \sigma \rrbracket^{\mathrm{Rel}}(z, w) \Longrightarrow \llbracket \sigma \rrbracket^{\mathrm{Rel}}(x, w)$$

(ii) *The induced relations on computation terms $\sim_\sigma$ is an equivalence relation.*

(iii) *The union of all the $\sim_\sigma$ relations is a congruence relation on terms.*

Proof. The proof of (i) is by induction on the structure of σ, and we refer to Appendix A for further details. By (i), $\sim_\sigma$ is symmetric and zigzag-closed, and by Proposition 4.4 it is reflexive. These properties imply transitivity, and so we conclude that it is an equivalence relation. Item (iii) is a consequence of compositionality of the interpretation. □

The relation $[\![\sigma]\!]^{\mathrm{Rel}}$ does not seem to be transitive, and this seems to be a general fact for relations constructed using relational reasoning for local store.

We end this section by showing that $\sim_\sigma$ is an approximation of contextual equivalence. To do this we first need to define contextual equivalence. A $(\Theta, \Gamma, \sigma) - \mathbf{b}$ context is a term $C[-]$ with a hole such that whenever $\Theta \mid \Gamma \vdash^c M : \sigma$ then $- \mid - \vdash^c C[M] : \mathbf{b}$ is well typed. Two open terms in the same context are contextually equivalent, written $\Theta \mid \Gamma \vdash^c M \equiv N : \sigma$ if $[\![C[M]]\!] = [\![C[N]]\!]$ for all $(\Theta, \Gamma, \sigma) - \mathbf{b}$ contexts $C[-]$.

Lemma 4.7 *Suppose $T^{\mathrm{Rel}}(eq_X)(f, g)$ for some set X and suppose f, g both have empty support. Then $f = g$.*

See Appendix B for a proof of Lemma 4.7.

Theorem 4.8 *If $\Theta \mid \Gamma \vdash^c M, N : \sigma$ and $M \sim_\sigma N$ then $\Theta \mid \Gamma \vdash^c M \equiv N : \sigma$.*

Proof. Suppose $C[-]$ is a $(\Theta, \Gamma, \sigma) - \mathbf{b}$ context. By item (iii) of Theorem 4.6 $C[M] \sim_{\mathbf{b}} C[N]$, which by Lemma 4.7 implies $[\![C[M]]\!] = [\![C[N]]\!]$. □

5 Algebraic operations in the relational model

We now show how the algebraic operations of Section 3.1 preserve relations on store by using the relational model. At the moment, we are missing one ingredient for this: a relational interpretation of the atom abstraction $[\mathbb{A}_{(-)}]X$ from nominal sets. To this end define $[\mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}}](X, Y, Q)$ to be the relation on $([\mathbb{A}^{\mathrm{Rel}}_{\mathrm{dom}(\mathbf{r})}]X, [\mathbb{A}^{\mathrm{Rel}}_{\mathrm{cod}(\mathbf{r})}]Y)$ relating (x, y) if $Q(x @ \mathbf{a_r}, y @ \mathbf{a_r})$ for all fresh $\mathbf{a_r}$.

Theorem 5.1 *Let (X, Y, Q) be an object in* **Rel** *and $\mathbf{r} = (\mathbf{b}, \mathbf{b}', R)$ be an element in Σ^{Rel}. Let* lookup, update, new *be as defined in Section 3.1. Then*

$$(\mathrm{update}_{X,\mathbf{b}}, \mathrm{update}_{Y,\mathbf{b}'}) \colon \mathbb{K}^Q_{\mathrm{Rel}} \to \mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}} \to_{\mathrm{fs}} \mathbf{r} \to_{\mathrm{fs}} \mathbb{K}^Q_{\mathrm{Rel}}$$
$$(\mathrm{lookup}_{X,\mathbf{b}}, \mathrm{lookup}_{Y,\mathbf{b}'}) \colon (\mathbf{r} \to_{\mathrm{fs}} \mathbb{K}^Q_{\mathrm{Rel}}) \to \mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}} \to_{\mathrm{fs}} \mathbb{K}^Q_{\mathrm{Rel}}$$
$$(\mathrm{new}_{X,\mathbf{b}}, \mathrm{new}_{Y,\mathbf{b}'}) \colon [\mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}}]\mathbb{K}^Q_{\mathrm{Rel}} \to \mathbf{r} \to_{\mathrm{fs}} \mathbb{K}^Q_{\mathrm{Rel}}$$

Proof. We just show the case of new. Suppose $([\mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}}]\mathbb{K}^Q_{\mathrm{Rel}})(f, g)$, $\mathbf{r}(v, v')$ and $Q(x, y)$. Write A for the union of the supports of f, g, x, y. We must show that if s, s' are stores satisfying $\mathbf{r}(s(\mathbf{a_r}), s'(\mathbf{a_r}))$ for all $\mathbf{a_r} \in A$, then

$$\mathrm{fresh}\, \mathbf{a}.\ (f @ \mathbf{a})(x)(s[\mathbf{a} \mapsto v]) = \mathrm{fresh}\, \mathbf{a}'.\ (g @ \mathbf{a}')(y)(s'[\mathbf{a}' \mapsto v'])\,. \tag{10}$$

The requirements on $\mathbf{a}$, $\mathbf{a}'$ in the equation above is that they are fresh (i.e., not in A) elements of $\mathbb{A}^{\mathrm{Rel}}$ mapped by the sorting $\mathbb{A}^{\mathrm{Rel}} \to \Sigma^{\mathrm{Rel}}$ to relations whose domain and codomain are respectively $\mathbf{b}$ and $\mathbf{b}'$. As long as these requirements are satisfied, the elements on both sides of (10) are independent of the concrete choice of atoms.

This means that we are allowed to choose some fresh $\mathbf{a_r}$ and use that for both atoms above. Since $([\mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}}]\mathbb{K}^Q_{\mathrm{Rel}})(f, g)$ and $Q(x, y)$ also $\mathbb{K}_{\mathrm{Rel}}((f @ \mathbf{a_r})(x), (g @ \mathbf{a_r})(y))$.

Moreover, the support of both $(f@\mathbf{a}_\mathbf{r})(x)$ and $(g@\mathbf{a}_\mathbf{r})(y)$ is contained in $A \cup \{\mathbf{a}_\mathbf{r}\}$, and

$$\mathbf{r}'(s[\mathbf{a}_\mathbf{r} \mapsto v](\mathbf{a}'_{\mathbf{r}'}), s'[\mathbf{a}_\mathbf{r} \mapsto v'](\mathbf{a}'_{\mathbf{r}'}))$$

for all $\mathbf{a}'_{\mathbf{r}'} \in A \cup \{\mathbf{a}_\mathbf{r}\}$. So $(f@\mathbf{a}_\mathbf{r})(x)(s[\mathbf{a}_\mathbf{r} \mapsto v]) = (g@\mathbf{a}_\mathbf{r})(y)(s'[\mathbf{a}_\mathbf{r} \mapsto v'])$ which proves (10). □

If we instantiate the case for new with $Q = \mathbb{K}^{eq_X}_{\mathrm{Rel}}$ we get the parametricity principle advertised in the introduction. One way to read the relation $[\mathbb{A}^{\mathrm{Rel}}_\mathbf{r}](T^{\mathrm{Rel}}[\![\sigma]\!]^{\mathrm{Rel}})$ is that it relates two computations with bound cell names, if the computations preserve the relation $\mathbf{r}$ between the contents of the abstracted cells. The statement for new in Theorem 5.1 is then the relational reasoning principle for local store. To see this, suppose that $a\colon \mathbf{b} \mid - \vdash^c M : \sigma$ and $a\colon \mathbf{b}' \mid - \vdash^c N : \sigma$ are such that $[\mathbb{A}^{\mathrm{Rel}}_\mathbf{r}](T^{\mathrm{Rel}}([\![\sigma]\!]^{\mathrm{Rel}}))((\mathbf{a}.\,[\![M]\!]_{(a\mapsto\mathbf{a})}), (\mathbf{a}.\,[\![N]\!]_{(a\mapsto\mathbf{a})}))$, and $\mathbf{r}([\![V]\!], [\![V']\!])$. Then also

$$T^{\mathrm{Rel}}([\![\sigma]\!]^{\mathrm{Rel}})(\mathrm{new}_{\mathbf{b},[\![\sigma]\!]}((\mathbf{a}.\,[\![M]\!]_{(a\mapsto\mathbf{a})}))([\![V]\!]), \mathrm{new}_{\mathbf{b}',[\![\sigma]\!]}((\mathbf{a}.\,[\![N]\!]_{(a\mapsto\mathbf{a})})([\![V']\!])))$$

i.e.,

$$\mathrm{let}\, a\colon \mathbf{b} = \mathrm{new}\,(V)\,\mathrm{in}\, M \sim \mathrm{let}\, a\colon \mathbf{b}' = \mathrm{new}\,(V')\,\mathrm{in}\, N\,,$$

which by Theorem 4.8 implies contextual equivalence.

Another consequence of Theorem 5.1 is that we can use the relational model to interpret not only the Fine-Grained CBV calculus with base types in Σ as we saw in Section 4, but also the larger Fine-Grained CBV with base types in Σ^{Rel}. Each base type in Σ^{Rel} is interpreted as itself, and the store access operations are interpreted using the algebraic operations

$$\begin{aligned}\mathrm{update}_{(X,Y,Q),\mathbf{r}} &= (\mathrm{update}_{X,\mathbf{b}}, \mathrm{update}_{Y,\mathbf{b}'})\\ \mathrm{lookup}_{(X,Y,Q),\mathbf{r}} &= (\mathrm{lookup}_{X,\mathbf{b}}, \mathrm{lookup}_{Y,\mathbf{b}'})\\ \mathrm{new}_{(X,Y,Q),\mathbf{r}} &= (\mathrm{new}_{X,\mathbf{b}}, \mathrm{new}_{Y,\mathbf{b}'})\,.\end{aligned}$$

Rather than considering algebraic operations, we could also consider generic effects, which also preserve relations. For example, the generic effect corresponding to new (6) satisfies

$$(\mathrm{alloc}_{\mathbf{b},\otimes_i \mathbb{A}_{\mathbf{b}_i}}, \mathrm{alloc}_{\mathbf{b}',\otimes_i \mathbb{A}_{\mathbf{b}'_i}})\colon \otimes_i \mathbb{A}_{\mathbf{r}_i} \times [\![\mathbf{r}]\!] \to T^{\mathrm{Rel}}((\otimes_i \mathbb{A}_{\mathbf{r}_i}) \otimes \mathbb{A}_\mathbf{r})\,.$$

6 Examples

To illustrate how the relational model captures relational reasoning for local store we show how to prove contextual equivalence of the counters defined in Section 2.

In the example of the counters, the signature Σ consists of the single type Int interpreted as the set $\mathbb{Z}$ of integers. We write $\mathrm{counter}_{\mathrm{up}}$ and $\mathrm{counter}_{\mathrm{down}}$ for the two counters. Consider the relation $R\colon \mathrm{Rel}(\mathbb{Z}, \mathbb{Z})$ relating x, z iff $x = -z$, and consider the four functions $f_d, f_c, g_d, g_c\colon \mathbb{Z} \times \mathbb{Z} \to \mathbb{Z}$ defined as follows

$$f_d(x,z) = z + x \qquad f_c(x,z) = z - x \qquad g_d(x,z) = z + x \qquad g_c(x,z) = x - z\,.$$

The pair $f = (f_d, f_c)$ defines a morphism $eq_{\mathbb{Z}} \times R \to R$ and the pair $g = (g_d, g_c)$ defines a morphism $eq_{\mathbb{Z}} \times R \to eq_{\mathbb{Z}}$ in **Rel**.

Consider a third counter $\mathrm{counter}_{\mathrm{rel}}$ defined in Fine-Grained-CBV over Σ^{Rel} as

$$\mathrm{let}\, a\colon R = \mathrm{new}\,((0,0))\,\mathrm{in}\,\mathrm{return}\,(\lambda x\colon eq_{\mathrm{Int}}.\,\mathrm{read}_a\,\mathrm{as}\,z.\,a := f(x,z); \mathrm{return}\,(g(x,z)))$$

Interpreting $\mathrm{counter}_{\mathrm{rel}}$ in the relational model gives $([\![\mathrm{counter}_{\mathrm{up}}]\!], [\![\mathrm{counter}_{\mathrm{down}}]\!])$ and so by well definedness of the interpretation in the relational model $\mathrm{counter}_{\mathrm{up}} \sim \mathrm{counter}_{\mathrm{down}}$. By Theorem 4.8 we conclude that $\mathrm{counter}_{\mathrm{up}}$ and $\mathrm{counter}_{\mathrm{down}}$ are contextually equivalent.

For a very different kind of example, consider the two computation terms

$$\begin{aligned} \mathrm{p}_1 &= \mathrm{let}\, a\colon \mathrm{Int} = \mathrm{new}\,(0)\,\mathrm{in}\,\mathrm{return}\,\lambda x\colon \mathrm{ref\ Int}.\,(x == a) \\ \mathrm{p}_2 &= \mathrm{return}\,\lambda x\colon \mathrm{ref\ Int}.\,\mathrm{return}\,(\mathrm{false})\,. \end{aligned}$$

Both these have type $\mathrm{ref\ Int} \to \mathrm{Bool}$. This example not only requires $\mathrm{Int}, \mathrm{Bool}$ to be base types, but also requires a comparison operator on reference. The latter, however, can be defined using equality on integers as

$$\lambda x\colon \mathrm{ref\ Int}.\,\mathrm{return}\,(\lambda y\colon \mathrm{ref\ Int}.\,\mathrm{read}_x\,\mathrm{as}\,z.\,y := z+1; \mathrm{read}_x\,\mathrm{as}\,w.\,\mathrm{return}\,\neg(w == z))$$

The computations p_1, p_2 should intuitively be equivalent because the first can never leak the locally created variable a, and so a can never occur as input to the function. We can give a formal argument for contextual equivalence using the relational model as follows.

Since p_2 equals $\mathrm{let}\, a\colon \mathrm{Int} = \mathrm{new}\,(0)\,\mathrm{in}\,\mathrm{return}\,(\lambda x\colon \mathrm{ref\ Int}.\,\mathrm{return}\,(\mathrm{false}))$ by the garbage collection axiom, it suffices to show that

$$([\![a\colon \mathrm{Int} \mid x\colon \mathrm{Int} \vdash^c x == a\colon \mathrm{Bool}]\!], [\![a\colon \mathrm{Int} \mid x\colon \mathrm{Int} \vdash^c \mathrm{return}\,(\mathrm{false})\colon \mathrm{Bool}]\!])$$

define a map of type $\mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}} \times \mathbb{A}^{\mathrm{Rel}}_{eq_{\mathbb{Z}}} \to T^{\mathrm{Rel}}(eq_{[\![\mathrm{Bool}]\!]})$ in **Rel** for some relation $\mathbf{r}$ on integers containing $(0,0)$. Recall that $\mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}}(\mathbf{a}, \mathbf{a}')$ holds iff $\mathbf{a} = \mathbf{a}'$ is some atom of sort $\mathbf{r}$, so verifying this boils down to showing that we can choose $\mathbf{r}$ in such a way that $[\![x == a]\!]_{(a \mapsto \mathbf{a_r}, x \mapsto \mathbf{a}'_{eq_{\mathbb{Z}}})}$ is $\eta(\mathrm{false}) \in T([\![\mathrm{Bool}]\!])$, for all $\mathbf{a_r}, \mathbf{a}'_{eq_{\mathbb{Z}}}$ where η is the unit of T. But in fact, if we choose $\mathbf{r}$ different from the identity relation, then this will always hold, because $\mathbf{a_r}$, $\mathbf{a}'_{eq_{\mathbb{Z}}}$ will range over the disjoint sets $\mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}}, \mathbb{A}^{\mathrm{Rel}}_{eq_{\mathbb{Z}}}$.

7 Towards more advanced relations on store

Up to now, the relations on store that we have considered have been the simplest possible: relating two stores if the contents of their cells are related pointwise. An obvious next step is to consider relations relating contents of several cells, such as

$$\{(s, s') \mid s(\mathbf{a}) + s(\mathbf{a}') = s'(\mathbf{a})\}$$

In this section we sketch how one can construct a relational model for reasoning with these kinds of relations on store using slightly more advanced atom sortings. The first step is to change the sorts in the relational model to relations on vectors of base types

$$\Sigma^{\mathrm{Rel}} = \{(\overline{\mathbf{b}}, \overline{\mathbf{b}'}, R) \mid \overline{\mathbf{b}}, \overline{\mathbf{b}'} \in \Sigma^*, R\colon \mathrm{Rel}(\llbracket\overline{\mathbf{b}}\rrbracket, \llbracket\overline{\mathbf{b}'}\rrbracket),\ R \text{ non-empty}\}\,.$$

where $\llbracket\overline{\mathbf{b}}\rrbracket$ is defined to be the product $\prod_i \llbracket\mathbf{b}_i\rrbracket$. Now consider the sets of atoms defined as

$$\mathbb{A}^{\mathrm{dom}} = \coprod_{(\overline{\mathbf{b}},\overline{\mathbf{b}'},R)\in\Sigma^{\mathrm{Rel}}} \coprod_{0\le i\le|\overline{\mathbf{b}}|} \mathbb{A} \qquad \mathbb{A}^{\mathrm{Rel}} = \Sigma^{\mathrm{Rel}} \times \mathbb{A}$$
$$\mathbb{A}^{\mathrm{cod}} = \coprod_{(\overline{\mathbf{b}},\overline{\mathbf{b}'},R)\in\Sigma^{\mathrm{Rel}}} \coprod_{0\le i\le|\overline{\mathbf{b}'}|} \mathbb{A}$$

The projection defines a sorting $\mathbb{A}^{\mathrm{Rel}} \to \Sigma^{\mathrm{Rel}}$, and there are also sortings $\mathrm{dom}\colon \mathbb{A}^{\mathrm{dom}} \to \Sigma$ and $\mathrm{cod}\colon \mathbb{A}^{\mathrm{cod}} \to \Sigma$ defined as $\mathrm{dom}((\overline{\mathbf{b}}, \overline{\mathbf{b}'}, R), i, \mathbf{a})) = \mathbf{b}_i$ and $\mathrm{cod}((\overline{\mathbf{b}}, \overline{\mathbf{b}'}, R), i, \mathbf{a})) = \mathbf{b}'_i$. There is a pair of homomorphisms

$$\mathbf{Perm}(\mathbb{A}^{\mathrm{dom}} \to \Sigma) \leftarrowtail \mathbf{Perm}(\mathbb{A}^{\mathrm{Rel}} \to \Sigma) \twoheadrightarrow \mathbf{Perm}(\mathbb{A}^{\mathrm{cod}} \to \Sigma)$$

e.g., the homorphism on the left maps the transposition $(((\overline{\mathbf{b}}, \overline{\mathbf{b}'}, R), \mathbf{a})\,((\overline{\mathbf{b}}, \overline{\mathbf{b}'}, R), \mathbf{a}'))$ to the permutation

$$(((\overline{\mathbf{b}}, \overline{\mathbf{b}'}, R), 1, \mathbf{a})\,((\overline{\mathbf{b}}, \overline{\mathbf{b}'}, R), 1, \mathbf{a}')) \circ \ldots \circ (((\overline{\mathbf{b}}, \overline{\mathbf{b}'}, R), n, \mathbf{a})\,((\overline{\mathbf{b}}, \overline{\mathbf{b}'}, R), n, \mathbf{a}'))$$

where n is the length of $\overline{\mathbf{b}}$. This pair of homorphisms induces a pair of functors

$$\mathbf{Nom}_{[\mathrm{dom}\colon \mathbb{A}^{\mathrm{dom}}\to\Sigma]} \twoheadrightarrow \mathbf{Nom}_{[\mathbb{A}^{\mathrm{Rel}}\to\Sigma^{\mathrm{Rel}}]} \leftarrowtail \mathbf{Nom}_{[\mathrm{cod}\colon \mathbb{A}^{\mathrm{cod}}\to\Sigma]}\,,$$

and with these in hand one can construct the category **Rel** exactly as before.

As we saw in Section 5, the key ingredient needed for expressing the parametricity principle for allocation was the atom abstraction in **Rel**. The definition can be generalised to the setting of the current section as follows. Given $\mathbf{r} = (\overline{\mathbf{b}}, \overline{\mathbf{b}'}, R)$ in Σ^{Rel} and (X, Y, Q) in **Rel**, we can defined the relation

$$[\mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}}]Q \subseteq [\mathbb{A}^{\mathrm{dom}}_{\overline{\mathbf{b}}}]X \times [\mathbb{A}^{\mathrm{cod}}_{\overline{\mathbf{b}'}}]Y$$

where

$$[\mathbb{A}^{\mathrm{dom}}_{\overline{\mathbf{b}}}]X = [\mathbb{A}^{\mathrm{dom}}_{\mathbf{b}_1}] \ldots [\mathbb{A}^{\mathrm{dom}}_{\mathbf{b}_n}]X$$
$$[\mathbb{A}^{\mathrm{cod}}_{\overline{\mathbf{b}'}}]Y = [\mathbb{A}^{\mathrm{cod}}_{\mathbf{b}'_1}] \ldots [\mathbb{A}^{\mathrm{cod}}_{\mathbf{b}'_m}]Y$$

by relating (x, y) iff for all fresh $(\mathbf{r}, \mathbf{a})$,

$$Q(x@\,(\mathbf{r}, 1, \mathbf{a}) \ldots @\,(\mathbf{r}, n, \mathbf{a}), y@\,(\mathbf{r}, 1, \mathbf{a}) \ldots @\,(\mathbf{r}, m, \mathbf{a}))\,.$$

The relation $[\mathbb{A}^{\mathrm{Rel}}_{\mathbf{r}}](T^{\mathrm{Rel}}\llbracket\sigma\rrbracket^{\mathrm{Rel}})$ relates two computations with bound sequences of cell names, if the computations preserve the relation $\mathbf{r}$ between the contents of

the abstracted cells. Using an appropriate generalisation of Theorem 5.1 involving vectors of update, lookup and new operations on each side, one can use this to prove soundness of relational reasoning using relations between vectors of cells. We leave further details to future publications.

Acknowledgement

I thank Nick Benton, Lars Birkedal, Andy Pitts, Alex Simpson, Ian Stark and Sam Staton for helpful discussions.

References

[1] N. Benton and B. Leperchey. Relational reasoning in a nominal semantics for storage. In *TLCA*, volume 3461 of *LNCS*, pages 86–101. Springer, 2005.

[2] N. Bohr and L. Birkedal. Relational reasoning for recursive types and references. In Naoki Kobayashi, editor, *APLAS*, volume 4279 of *Lecture Notes in Computer Science*, pages 79–96. Springer, 2006.

[3] Brian Dunphy and Uday S. Reddy. Parametric limits. In *LICS*, pages 242–251. IEEE Computer Society, 2004.

[4] J. Egger, R. E. Møgelberg, and A. Simpson. Enriching an effect calculus with linear types. In *CSL*, volume 5771 of *LNCS*, pages 240–254. Springer, 2009.

[5] M. J. Gabbay and A. M. Pitts. A new approach to abstract syntax with variable binding. *Formal Aspects of Computing*, 13:341–363, 2001.

[6] Paul Blain Levy. *Call By Push Value, a Functional/ Imperative Synthesis.* Kluwer, December 2003.

[7] P.B. Levy, J. Power, and H. Thielecke. Modelling environments in call-by-value programming languages. *Information and Computation*, 185, 2003.

[8] E. Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.

[9] P. W. O'Hearn and J. C. Reynolds. From algol to polymorphic linear lambda-calculus. *Jrnl. A.C.M.*, 47(1):167–223, January 2000.

[10] P. W. O'Hearn and R. D. Tennent. Parametricity and local variables. *Jrnl. A.C.M.*, 42(3):658–709, 1995.

[11] F. J Oles. *A category-theoretic approach to the semantics of programming languages.* PhD thesis, Syracuse University, Syracuse, N.Y, 1982.

[12] A. M. Pitts and I. D. B. Stark. Operational reasoning for functions with local state. In *Higher Order Operational Techniques in Semantics*, pages 227–273. Cambridge University Press, 1998.

[13] G. Plotkin. Call-by-name, call-by-value, and the λ-calculus. *Theoret. Comp. Sci.*, 1:125–159, 1975.

[14] G. Plotkin and J. Power. Notions of computation determine monads. In *FOSSACS*, volume 2620 of *LNCS*, pages 342–356, 2002.

[15] G. D. Plotkin and J. Power. Algebraic operations and generic effects. *Appl. Categ. Structures*, 11(1):69–94, 2003.

[16] John C. Reynolds. The essence of Algol. In *Proceedings of the 1981 International Symposium on Algorithmic Languages*, pages 345–372. North-Holland, 1981.

[17] M. R. Shinwell and A. M. Pitts. On a monadic semantics for freshness. *Theor. Comput. Sci*, 342(1):28–55, 2005.

[18] S. Staton. Completeness for algebraic theories of local state. In *FOSSACS*, volume 6014 of *LNCS*, pages 48–63. Springer, 2010.

A Proof of Theorem 4.6

We first give the definition of the isomorphisms $[\![\sigma]\!]^{\mathrm{dom}} \cong [\![\sigma]\!]^{\mathrm{cod}}$. Abusing notation, we shall simply write $(-)^{\mathrm{op}}$ for both directions of this isomorphism.

The needed isomorphism

$$[\![\mathrm{ref}\ \mathbf{b}]\!]^{\mathrm{dom}} = \mathbb{A}^{\mathrm{Rel}}_{\mathrm{dom}(\mathbf{b})} \to [\![\mathrm{ref}\ \mathbf{b}]\!]^{\mathrm{cod}} = \mathbb{A}^{\mathrm{Rel}}_{\mathrm{cod}(\mathbf{b})}$$

(notation as in (7, 8)) is simply a restriction of the $(-)^{\mathrm{op}}$ assumed to exist. For the case of base types $\mathbf{b}$, we can take the needed isomorphism to be simply the identity.

For the rest of the cases we need an isomorphism

$$\mathbb{K}_{\mathrm{dom}} \cong \mathbb{K}_{\mathrm{cod}}$$

This is defined to map k to k^{op} where $k^{\mathrm{op}}(s) = k(s \circ (-)^{\mathrm{op}})$. It is convenient to write s^{op} for $s \circ (-)^{\mathrm{op}}$. The rest of the cases of $[\![\sigma]\!]^{\mathrm{dom}} \cong [\![\sigma]\!]^{\mathrm{cod}}$ are simply defined by lifting the isomorphism to function spaces in the usual way by defining $f^{\mathrm{op}}(x) = (f(x^{\mathrm{op}}))^{\mathrm{op}}$.

We must show that $(-)^{\mathrm{op}} \circ [\![\sigma]\!]^{\mathrm{Rel}}$ is a symmetric and zigzag-closed relation on $[\![\sigma]\!]^{\mathrm{dom}}$. This is done by induction on σ. The case of base types is trivial. In the case of references the relation $(-)^{\mathrm{op}} \circ [\![\mathrm{ref}\ \mathbf{b}]\!]^{\mathrm{Rel}}$ is the relation $\{(\mathbf{a}_{eq_{\mathbf{b}}}, (\mathbf{a}_{eq_{\mathbf{b}}})^{\mathrm{op}}) \mid \mathbf{a}_{eq_{\mathbf{b}}} \in \mathbb{A}^{\mathrm{Rel}}\}$, which is clearly symmetric and zigzag-closed.

We show that $(-)^{\mathrm{op}} \circ \mathbb{K}_{\mathrm{Rel}}$ is a symmetric relation on $\mathbb{K}_{\mathrm{dom}}$. For this it suffices to show that $(k_d, k_c) \in \mathbb{K}_{\mathrm{Rel}}$ implies $(k_c^{\mathrm{op}}, k_d^{\mathrm{op}}) \in \mathbb{K}_{\mathrm{Rel}}$ for any $k_d \in \mathbb{K}_{\mathrm{dom}}, k_c \in \mathbb{K}_{\mathrm{cod}}$. So, assume that there exists some $A \subseteq_{\mathrm{fin}} \mathbb{A}^{\mathrm{Rel}}$ such that for all

$$\forall s_d, s_c.\, (\forall \mathbf{a}_{\mathbf{r}} \in A.\, \mathbf{r}(s_d(\mathbf{a}_{\mathbf{r}}), s_c(\mathbf{a}_{\mathbf{r}}))) \Longrightarrow k_d(s_d) = k_c(s_c) \tag{A.1}$$

we will show that for all s_d, s_c

$$\forall s_d, s_c.\, (\forall \mathbf{a}_{\mathbf{r}} \in A^{\mathrm{op}}.\, \mathbf{r}(s_d(\mathbf{a}_{\mathbf{r}}), s_c(\mathbf{a}_{\mathbf{r}}))) \Longrightarrow k_c^{\mathrm{op}}(s_d) = k_d^{\mathrm{op}}(s_c) \tag{A.2}$$

where $A^{\mathrm{op}} = \{\mathbf{a} \mid \mathbf{a}^{\mathrm{op}} \in A\}$. Note first that (A.2) can be reformulated as

$$\forall s_d, s_c.\, (\forall \mathbf{a}_{\mathbf{r}} \in A.\, \mathbf{r}^{\mathrm{op}}(s_d(\mathbf{a}_{\mathbf{r}}^{\mathrm{op}}), s_c(\mathbf{a}_{\mathbf{r}}^{\mathrm{op}}))) \Longrightarrow k_d^{\mathrm{op}}(s_d) = k_c^{\mathrm{op}}(s_c) \tag{A.3}$$

If (s_d, s_c) satisfy the hypothesis of (A.3) then $(s_c^{\mathrm{op}}, s_d^{\mathrm{op}})$ satisfy the hypothesis of (A.1), from which we conclude $k_d(s_c^{\mathrm{op}}) = k_c(s_d^{\mathrm{op}})$. Since $k_d(s_c^{\mathrm{op}}) = k_d^{\mathrm{op}}(s_c)$ and $k_c(s_d^{\mathrm{op}}) = k_c^{\mathrm{op}}(s_d)$, we conclude the proof of $(-)^{\mathrm{op}} \circ \mathbb{K}_{\mathrm{Rel}}$ being symmetric.

To show that $(-)^{\mathrm{op}} \circ \mathbb{K}_{\mathrm{Rel}}$ is zigzag-closed it suffices to show that $\mathbb{K}_{\mathrm{Rel}}$ is. If A witnesses $\mathbb{K}_{\mathrm{Rel}}(k_d, k_c)$, A' witnesses $\mathbb{K}_{\mathrm{Rel}}(k'_d, k_c)$ and A'' witnesses $\mathbb{K}_{\mathrm{Rel}}(k'_d, k'_c)$ then $A \cup A' \cup A''$ witnesses $\mathbb{K}_{\mathrm{Rel}}(k_d, k'_c)$.

The rest of the proof follows from the following easy lemma.

Lemma A.1 *Suppose $R \subseteq X \times Y$ and $S \subseteq W \times Z$ are relations on sets. Consider the relation $R \to S \subseteq (X \to W) \times (Y \to Z)$ relating f, g if $S(f(x), g(y))$ for all x, y such that $R(x, y)$. Then*

- *If R and S are symmetric so is $R \to S$.*
- *If S is zigzag-closed, so is $R \to S$.*

B Proof of Lemma 4.7

For the proof we need the following definition and lemma. We say that an element x has support on the diagonal, if all $\mathbf{a}$ in the support of x are mapped by p to an identity relation.

Lemma B.1 *Suppose $k_d, k_c \in \mathbb{K}_{\mathrm{dom}}$ both have support on the diagonal. Then $(k_d, k_c^{\mathrm{op}}) \in \mathbb{K}_{\mathrm{Rel}}$ iff $k_d = k_c$.*

Proof. Suppose first $(k_d, k_c^{\mathrm{op}}) \in \mathbb{K}_{\mathrm{Rel}}$ and $s \in \mathbb{S}_{\mathrm{dom}}$. We must show that $k_d(s) = k_c(s)$. Note first that since k_c has support on the diagonal, so has k_c^{op}. Since $\mathbf{a}_{eq_{\mathbf{b}}}^{\mathrm{op}} = \mathbf{a}_{eq_{\mathbf{b}}}$, also $s(\mathbf{a}_{eq_{\mathbf{b}}}) = s^{\mathrm{op}}(\mathbf{a}_{eq_{\mathbf{b}}})$ for any $\mathbf{a}_{eq_{\mathbf{b}}}$, so the requirement of support on the diagonal implies the condition

$$\forall \mathbf{a}_{\mathbf{r}} \in \mathrm{supp}(k_d) \cup \mathrm{supp}(k_c^{\mathrm{op}}).\, \mathbf{r}(s(\mathbf{a}_{\mathbf{r}}), s^{\mathrm{op}}(\mathbf{a}_{\mathbf{r}}))$$

to be satsified. Now, by Lemma 4.1 we conclude $k_d(s) = k_c^{\mathrm{op}}(s^{\mathrm{op}})$. Since by definition $k_c^{\mathrm{op}}(s^{\mathrm{op}}) = k_c(s)$ we conclude $k_d = k_c$.

On the other hand, suppose we are given k with support on the diagonal and s_d, s_c such that

$$\forall \mathbf{a}_{\mathbf{r}} \in \mathrm{supp}(k) \cup \mathrm{supp}(k^{\mathrm{op}}).\, \mathbf{r}(s_d(\mathbf{a}_{\mathbf{r}}), s_c(\mathbf{a}_{\mathbf{r}}))\,.$$

We must show that $k(s_d) = k^{\mathrm{op}}(s_c)$. Since k has support on the diagonal, the condition says that s_d and s_c agree on the support of k, so $k(s_d) = k(s_c)$, by Lemma 3.1. Since any store s agrees with s^{op} on the support of k, Lemma 3.1 also implies that $k = k^{\mathrm{op}}$, and so we conclude $k(s_d) = k^{\mathrm{op}}(s_c)$. □

Suppose $T^{\mathrm{Rel}}(eq_X)(f, g)$ for some set X and suppose f, g both have empty support. Precisely, f, g are both elements of $\mathbb{K}_{\mathrm{dom}}^{\mathbb{K}_{\mathrm{dom}}^X}$ and the assumption of $T^{\mathrm{Rel}}(eq_X)(f, g)$ should be read up to the isomorphism constructed in Section A. We will show that $f(k) = g(k)$ for all k.

The continuation k may not have support on the diagonal, but there is a permutation π such that $\pi \cdot k$ does. Since f, g both have empty support it suffices to show that $f(\pi \cdot k) = g(\pi \cdot k)$.

The assumption says exactly that $(f, g^{\mathrm{op}})\colon (eq_X \to \mathbb{K}_{\mathrm{Rel}}) \to \mathbb{K}_{\mathrm{Rel}}$. Note that for any x, $(\pi \cdot k)(x)$ has support on the diagonal, and so by Lemma B.1

$$(eq_X \to \mathbb{K}_{\mathrm{Rel}})(\pi \cdot k, (\pi \cdot k)^{\mathrm{op}})\,.$$

So $(f(\pi \cdot k), g^{\mathrm{op}}(\pi \cdot k)^{\mathrm{op}}) \in \mathbb{K}_{\mathrm{Rel}}$. Since both these elements have support on the diagonal,

$$f(\pi \cdot k) = (g^{\mathrm{op}}(\pi \cdot k)^{\mathrm{op}})^{\mathrm{op}}\,.$$

Finally, since $(g^{\mathrm{op}}(\pi \cdot k)^{\mathrm{op}})^{\mathrm{op}} = g(\pi \cdot k)$ we conclude $f = g$.